

Spam: Technologies and Policies

White Paper
November 2003

Joshua Goodman
Microsoft Research

For more information, contact:

Samantha McManus
Waggener Edstrom
(425) 638-7000
samantham@wagged.com

Rapid Response Team
Waggener Edstrom
(503) 443-7070

© 2003 Microsoft Corp. All rights reserved.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not

be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS," "WITH ALL FAULTS," AND "AS AVAILABLE," AND THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH YOU. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. MICROSOFT DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS, STATUTORY AND IMPLIED, INCLUDING WITHOUT LIMITATION (1) WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, WORKMANLIKE EFFORT, ACCURACY, TITLE, NO LIENS AND NON-INFRINGEMENT, (2) WARRANTIES ARISING THROUGH COURSE OF DEALING OR USAGE OF TRADE, AND (3) WARRANTIES THAT ACCESS TO OR USE OF ANY MICROSOFT PRODUCTS OR SERVICES MENTIONED HEREIN WILL BE UNINTERRUPTED OR ERROR-FREE. IN NO EVENT SHALL MICROSOFT BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL OR PUNITIVE DAMAGES ARISING OUT OF, BASED ON, OR RESULTING FROM THE INFORMATION OR SERVICES DESCRIBED HEREIN OR YOUR USE OF THAT INFORMATION OR THOSE SERVICES, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF USE, DATA OR PROFITS, WITH THE DELAY OR INABILITY TO USE ANY MICROSOFT SERVICE, OR OTHERWISE ARISING OUT OF THE USE OF THE INFORMATION HEREIN, WHETHER BASED ON CONTRACT, TORT, NEGLIGENCE, STRICT LIABILITY OR OTHERWISE, EVEN IF MICROSOFT HAS BEEN ADVISED OF THE POSSIBILITY OF DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Microsoft, the Office logo, Outlook, Hotmail and MSN are either registered trademarks or trademarks of Microsoft Corp. in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Disclaimer: This document represents the personal views of the author, which are not necessarily the same as the views of Microsoft Corp.

Acknowledgements: This document is based on a huge number of sources, including but not limited to discussions with, e-mail from and documents written by, in no particular order: Robert Rounthwaite, Cynthia Dwork, Bob Atkinson, David Heckerman, Elissa Murphy, John Mehr, Nathan Howell, Micah Rupersburg, Bryan Starbuck, Adam Back, Manav Mishra, Ryan Hamlin, Anthony Penta, John Deurbrouck, Khaja Ahmed, Jim Lyon, Brian Arbogast, Harry Katz, Kevin Schofield, Diane McDade, Dean Slawson, Ted Wobber, Rick Holzli, Pablo Stern, Eliot Gillum, Andrew Birrell and Mike Burrows. Special thanks to Ira Rubinstein for extensive help editing this document.

INTRODUCTION.....	1
TECHNOLOGIES FOR BLOCKING SPAM.....	1
Machine Learning Systems	1
Matching Systems	2
Block Lists.....	3
Safe Lists	4
Challenge-Response and Micropayment or Postage Systems.....	4
Ephemeral or Disposable E-Mail Addresses	7
TECHNIQUES SPAMMERS USE	7
Open Relays.....	7
Open Proxies.....	7
Web Harvesting and Dictionary Attacks.....	8
INDUSTRY INITIATIVES	8
E-Mail Best-Practice Programs.....	9
Propagating Lists: Technical Issues.....	9
E-Mail Best-Practice Programs: Standards.....	9
Anti-Spoofing.....	10
CONCLUSIONS	10
Technical Details	11
Spoofting.....	11
Machine Learning Systems.....	13
Bayesian Filters.....	14
REFERENCES.....	16

Introduction

Spam is becoming an increasingly large problem. Some Internet service providers (ISPs) receive over a billion spam messages per day. Much of this mail is filtered before it reaches end users, but much is delivered. To end users, spam is mainly annoying and inconvenient, but in many cases, it is far worse. Spam is often fraudulent, such as the well-known “Nigerian” scam. Many users overlook legitimate e-mail because it is buried in a mountain of spam. All filtering technologies are imperfect, and occasionally delete a piece of legitimate mail before the mail reaches its intended recipient. Even worse, children routinely receive very explicit pornographic messages. Despite these many social harms and the efforts of technology vendors, ISPs and law enforcement agencies to combat spam, the problem is getting worse, not better. Spam appears to be increasing exponentially, with no limits in sight given the peculiar economics of spam: It costs close to nothing to send spam, yet spammers can make money even if their response rates are extremely low.

In this document, we briefly describe and evaluate possible solutions to spam. We do not believe that any single solution will suffice, nor that any solution is right for all users. Some users, such as salespeople or researchers, routinely receive e-mail from people they do not know, and it is important to them not to miss any legitimate e-mail. Others, such as children, may receive e-mail from a small number of people, and it may be more important that they only hear from this small number and never receive pornography than that a new person be able to reach them.

Spam is not a simple problem. Spammers use many techniques to send spam and to evade filters. For each technique, there are usually multiple counter-techniques. This document then is of necessity brief and simplified.

Technologies for Blocking Spam

There are many different technologies for blocking spam. The most important include machine learning systems, systems that match known spam, block lists and challenge-response systems. All of these systems may occasionally block legitimate e-mail; safe lists work synergistically with these systems to reduce the number of “false positives” — legitimate e-mail marked as spam. None of these systems is perfect: All miss some spam, and all catch some legitimate mail.

Machine Learning Systems

Machine learning systems for blocking spam were invented at Microsoft Research [4], and since then, many other people have also explored this technique. These systems use learning techniques such as neural networks, naive Bayes, or other techniques to block spam. Microsoft® spam filtering solutions use machine learning.

Machine learning systems are presented with a large amount of training data — minimally thousands of messages, but ideally millions — labeled as good mail or spam. They then learn to distinguish the good mail from the spam. They learn that certain words such as “click,” “free” and “wet” are indicative of spam, and words such as “tomorrow” and “weather” are indicative of good mail. They also factor in other properties of the message. For instance, messages with links and images are much more likely to be spam than messages with neither.

Machine learning systems have several good properties. Even messages they have never seen before can be labeled as spam or good. These systems typically produce a probability of mail being spam or good, so that an appropriate action can be taken: Mail that is definitely spam can be deleted, while mail that is probably spam can be put in a special folder, and mail that just might be spam can be marked as suspicious. Some machine learning filters, such as the filters shipping with MSN[®] 8 and forward, enable users to teach them their preferences as to what counts as spam, thus allowing more personalized filtering.

Machine learning systems are not without problems. Although it can be difficult to acquire enough data to train such systems, Microsoft has recently solved this problem with the help of a large number of volunteers who use Hotmail[®]. Spammers may successfully make their messages very similar to good messages, confusing such systems. Spammers may use very short messages with embedded images that are difficult to analyze. Still, we believe we can overcome these problems through sophisticated message analysis, and that machine learning systems represent one of the best short-term ways to combat spam. See the Appendix for a bit more explanation of the ways that machine learning systems work, and for a discussion of Bayesian filtering, a machine learning technique that has recently become popular for stopping spam.

Matching Systems

Matching systems attempt to match messages to known spam. For instance, some systems compile a list of messages reported by users to be spam. They then look for any new messages that match this known spam, and mark them as spam, too. Spammers have quickly caught on to these exact match techniques and have started randomizing their messages. For instance, many pieces of spam now include random character sequences at the end of the subject line. For example:

```
Re: most unlikely! ydxawxnduqzu wl
```

These random sequences are different for every message and thus defeat simple exact matching systems.

One alternative to exact matching systems is rule-based systems. These systems typically rely on a team of technicians who detect new incoming messages and write rules that attempt to capture the commonalities between these messages. Alternatively, complex automated systems may write the rules. Either way, this is not an easy task. If a new rule is too general, it will result in false positives, deleting good messages. If the rule is too specific, the spammer's randomization will defeat it. In addition, it is impossible for these systems to react instantly, often requiring several minutes to propagate new rule sets; spammers who change their messages more quickly than the technicians can write, validate and distribute new rules can get their messages through.

A different matching technique is so-called fuzzy hashing. These systems attempt to find the commonalities in messages and compute a large number (a hash) based on these commonalities. Any message with the same large number is presumably also spam. To defeat randomization techniques, these systems will, for instance, detect random characters at the end of the subject line and not include them in the fuzzy hash, as well as perhaps leaving out punctuation, etc. The fuzzier the system, the better it is at catching spam, but the more likely it is to catch good mail accidentally.

Matching systems can get examples of spam in two basic ways: user complaints and “honeypots.” User complaint systems can range from ad-hoc reports to “report spam” buttons. Honeypots (also called trap accounts, sentinels and many other names) are accounts that should never receive mail, such as newly created, virgin accounts (i.e., accounts that no one has ever used to send mail, subscribe to a newsletter, receive a delivery notification, etc.). Any mail that goes to these accounts is by definition unsolicited or spam.

We are skeptical about the long-term success of matching systems because messages can be randomized with almost infinite variety. In one method that we call the “mad libs” attack, almost every part of the message is different every time it is sent. Sentences are selected from random lists, and even the words in these sentences can be randomized (e.g., “free,” “no cost” or “no charge”.) Rule-based and fuzzy hashing systems are defeated by such techniques. We have already seen examples of spammers who use this method successfully. In addition, the number of unique rules or hashes needed to list all these variations can become very large, slowing e-mail servers substantially, and, as mentioned, propagation times are an issue as well.

Proponents of matching systems sometimes claim that their false positive rates are extremely low because they match only known spam. It is very difficult to measure false positive rates because they are primarily based on user complaints, and users rarely complain about mail they didn’t receive — they usually have no way to know they didn’t receive it! In our own experiments, we have found that matching systems have false positive rates comparable to, or worse than, machine learning systems.

Block Lists

Spammers must send their mail *from* somewhere. It turns out that the “From” line on messages is extremely easy to forge, as are most other parts of an e-mail message. However, the IP address that a message comes from — its Internet address — is nearly impossible to forge. (IP addresses are sets of four numbers, such as 101.202.73.2, that uniquely specify computers on the Internet.) Block lists (sometimes called blackhole lists) are lists of IP addresses that are recognized as originators of spam. These lists can be compiled in ways similar to the way matching systems work, using either user complaints or honeypots. Open proxies and open relays (described below) are often listed in IP block lists.

Block lists have been plagued by a number of problems. It has been difficult to keep the lists up to date quickly enough to reflect new information. Block lists also sometimes include innocent victims of spammers exploiting security holes, and these individuals or businesses often have trouble persuading the list owners to remove their IP addresses even after the security holes are fixed. Some purveyors of block lists have been overly aggressive in listing IP addresses and have even listed addresses of those who send no spam but who may aid spammers through lax enforcement on other IP addresses. Some IP addresses are sources of both spam and good mail, and block lists do not distinguish between them. All of these problems have led to lawsuits

against entities that create or maintain the lists.¹ Finally, block lists are typically implemented at the router level, meaning that individuals cannot bypass the block by using a safe list.

Safe Lists

One technology that works in conjunction with machine learning systems and matching systems is that of “safe lists” (sometimes also called “white lists”). Safe lists are lists of people who are recognized as senders of good e-mail. Typically, they are individual senders, but whole domains can be listed too. If a machine learning system marks a message as spam but the sender is on the user’s safe list, the message is allowed through. This helps mitigate false positives.

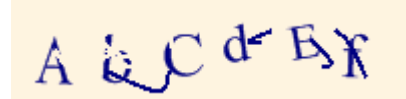
Some users even prefer an extreme variation on safe lists, known as “exclusive mode.” In this mode, only mail from users on the safe list is allowed into the user’s inbox. All other mail is put in a spam folder. Users can look through the spam folder to see if any mail was inadvertently categorized as spam. For some users, such as children, or those who do not receive mail from people they do not know, this may be a good option, albeit extreme. But this approach severely restricts the utility of e-mail as a communications medium. For example, it makes it very difficult to conduct e-commerce, since users can rarely predict what address an e-commerce site will use to send a receipt, or to receive e-mail from old friends or new business associates. In these cases, mail that the user expects, or would like to receive, will not be delivered; instead it will be lost, or at least delayed until the user looks into his or her spam folder and retrieves it.

Safe lists are not foolproof. It is extremely easy to forge the address of senders on the Internet. The mail protocol used on the Internet essentially allows anyone to impersonate anyone else. Many users place themselves or their friends on safe lists. This is part of the reason you may see spam apparently “from” yourself: a spammer is trying to use your safe list against you. In the Microsoft Research Machine Learning and Applied Statistics group, we sometimes see spam “from” other group members: Some clever spammers have noticed our e-mail addresses all together on Web pages and have assumed that we are probably on each others’ safe lists, or at least likely to open mail from each other. Later we discuss new standards to prevent such spoofing, but they will require much more time and industry cooperation.

Challenge-Response and Micropayment or Postage Systems

Challenge-response systems attempt to raise the cost of spamming. In these systems, when mail is not recognized as legitimate or otherwise appears to be spam, a “challenge” is given to the sender. Not until the sender answers the challenge does the recipient get the message. These challenges can be of three broad types: human, computational and monetary.

Human challenges, also known as human interactive proofs (HIPs), CAPTCHAs (so named by researchers as an acronym for completely automated public Turing test to tell computers and humans apart) or reverse Turing tests, may consist of a picture such as the following: [5]



The user must respond by typing in the letters in the picture.

¹ For example, the Mail Abuse Protection System LLC (MAPS) has been sued by Media3 Technologies LLC, Black Ice Software Inc., Harris Interactive and Yesmail.com.

Almost any human can solve this problem, but with a good challenge-response system, computers currently cannot.²

In computational challenges, a computer system must solve a puzzle [1][2][3]. To give an analogy, the recipient gives a jigsaw puzzle to the sender, with the pieces in random order. The sender must figure out the positions of the puzzle pieces and send the solution back to the recipient. The recipient's computer system verifies that the puzzle was correctly solved and then moves the message from the recipient's junk folder to their inbox. It is quick for the recipient to create the puzzle and quick for them to verify the solution but very time-consuming for the sender to solve. (The actual puzzle is a bit different: typically, the sender must find a number whose hash value is the same as a hash of the complete message or its headers.) Computational challenges have an advantage in that they can be automated: If there is a standard for computational challenges, then I can configure my computer to automatically solve any computational challenges to mail I have sent. I will then never be bothered by these challenges — I will send mail, your system will automatically challenge it, my system will automatically respond, and you will get the message. Of course, to prevent spammers from automatically solving these puzzles, the amount of computation needed must be kept very high, perhaps 30 seconds to five minutes. If the puzzle is too easy to solve — say that it could be solved in one second or five seconds — then spammers could afford to buy enough computers to solve all of these challenges, and continue sending spam.

Finally, there are micropayment systems. In these systems, for example, I agree to receive your mail on the condition that you send me a small sum of money, perhaps a penny or a nickel, through some form of online banking [6]. Perhaps I actually take your money only if your message is spam. The recipient of the message can receive the money, but this invites abuse by receivers, who may trick senders into sending them mail, challenging them and then keeping their payment. An alternative is for the money to go to a charity, or to the recipient's ISP.

One way to improve the efficiency of challenge-response systems is to combine them with other systems. For instance, they are typically combined with safe lists (see above.) People on the safe list are not challenged, and once someone has answered a challenge, she is automatically added to the safe list, so that she will not be challenged again. In addition, they can be combined with spam detectors (machine learning or a matching system). Only mail marked as spam by a filter, and from someone not on the safe list, is challenged. This combination means that far fewer people receive challenges (thereby minimizing unnecessary network traffic), but a small amount of spam may get through.

In some versions of challenge-response, such as the Penny Black Project, the challenges are “bankable.” In this version, one can get tokens of some sort, perhaps by performing computation, perhaps by solving human challenges and perhaps by paying money. These tokens are recorded in an account. The “challenge” is a request for a token from the sender's account. The token may be refundable, or may be removed from the sender's account only if the recipient says the message is spam. An advantage of bankable or refundable challenges is that they can be a stronger disincentive to spammers. I'm not willing to do an hourlong computational challenge, or solve a complex human challenge, or pay 25 cents to send you a message, but if my message is

² See <http://www.aladdin.cs.cmu.edu/hips/events/list.html> for a broad discussion of HIPs.

not spam, and I know you, I'm willing to risk a token that costs that much. Since you're probably a nice guy, you will refund the token to me when you find that my message isn't spam. But if you're a spammer and you try this, you will quickly spend a *lot* of money, more than you make by sending spam. Thus bankable systems let the cost to spammers be very high, while the cost to legitimate senders is near zero. On the other hand, this version is somewhat more complex. It requires either setting up a single worldwide bank (Who will establish and maintain it? Who will pay for it? Will it make or lose money?) or a network of banks, but then how do I know which banks to trust? It also requires putting a hold on the account for every message, to prevent someone with \$1 in his account from sending a million messages. This means an extra transaction (or pair of transactions) for every message sent on the Internet, which may increase costs substantially. Who will pay these costs?

In one variation on challenge-response system, the challenges are precomputed. Senders always perform a computation, or attach a token, or attach a refundable micropayment before they send a message. This avoids delays from the round-trip nature of challenges but may mean that senders perform unnecessary computations. Any system based purely on precomputation seems unworkable because those who do not precompute will not know that their messages are rejected. As an option in challenge-response systems, however, precomputation may be desirable.

Challenge-response systems suffer from a number of problems. Some human challenges we have seen are far too easy and could be automated by computers. For instance, those that simply ask multiple-choice questions can use a computer to guess at random, which will allow an excessive amount of spam to get through.³ There may be ways for spammers to hijack either people or computers. For instance, spammers can pay people to solve human challenges (perhaps high school students or people in countries with very low wages) or they may provide access to online content (pornography? free greeting cards? news archives?) in exchange for solving challenges. Worse, they may be able to subvert computers for use in solving computational challenges. Virus writers have already started using infected computers to send spam; this use is pretty quickly stopped because the sender's ISP contacts them or terminates their account. But if a virus solves computational challenges in the background, at low priority, not disrupting computer use, and if the answer is forwarded to another computer so that the solving computer cannot be located, then the owner of a subverted computer may never know that the computer is being used for this purpose and has little incentive to disinfect the computer even if notified. Another problem is that some poorly implemented challenge-response systems do not work well with mailing lists (they may challenge all mail sent to the list, or even send their challenge to the entire list.)

The biggest advantage of HIP and computation-based challenge-response systems is that they guarantee that any sufficiently motivated sender can get their mail through to the recipient. There is a lot to be said for this guarantee. Refundable micropayment systems have additional advantages, in that they are less costly for large legitimate senders, but are more complicated to implement. We think that challenge-response systems of various sorts are very promising, and they are an active area of research.

³ See [5] for an HIP designed by an expert in OCR systems that is unlikely to be breakable by computers in the near future.

Ephemeral or Disposable E-Mail Addresses

One approach to fighting spam that has received less attention but is worthy of mention is ephemeral e-mail addresses, also called disposable addresses. In this solution, senders give a different address to each person they correspond with. If they start receiving spam at that address, they can press a button to terminate it; all future mail to that address is bounced. I can give an address to Amazon.com, but if Amazon abuses this address, I terminate them with a single click.

Ephemeral addresses have a lot of potential, but there are a several difficulties. First, it may be difficult for someone I don't know to contact me. What address do I put on my business card? What address do I put on my Web page? Also, the addresses must be sufficiently long that they cannot be guessed by spammers. This generally means including a string of random letters and numbers so that the addresses are hard to remember. If I send a single message to two friends and have given each of them a different address, which "From" address do I use? Finally, many solutions to spam can be implemented in either client software (e.g., Microsoft Outlook[®]) or server software (e.g., Microsoft Exchange Server). Ephemeral e-mail addresses require changes to client and server (changes on the client, so that the user interface can be changed to include "blocking" buttons as well as buttons to generate new addresses, and changes on the server so that mail to different addresses all goes to the same inbox). Many users control their client software but not their server software, and requiring the double change means that only those users who change both client and server software can use this solution.

Techniques Spammers Use

Open Relays

For a number of reasons, spammers often try to obfuscate the point of origin of their mail. Sometimes their mail is fraudulent. Often, sending spam is a violation of their ISP's terms of service, and spam complaints result in account termination. Block lists (lists of IP addresses that send spam and should be blocked) are a problem for spammers.

One technique (but less common now) that spammers use for avoiding detection is open relays. These are e-mail servers that will send mail on behalf of someone else. Typically, an e-mail server for x.com will accept mail only for addresses at x.com. But some servers, open ones, will also accept mail for y.com; they will then contact the y.com server and forward the mail. Open relays exist for several reasons. Some people need to use relay servers because of firewalls and are not willing to restrict these servers to accept mail only from the intended users. Some older software, predating spam, was configured to be open by default. Those who create open relays almost never do so intentionally, or at least not with the intention of helping spammers. The IP addresses of these mail servers are typically added to IP block lists, meaning that legitimate mail does not get through; the maintainers of open relays thus have strong incentives to close these relays.

Open Proxies

Proxy servers are designed to help users of the Web who are behind firewalls; the proxy server forwards their Web requests around the firewall, which otherwise might block it. While proxy

servers are designed for Web requests, it turns out that in misconfigured servers, a rarely used command, HTTP CONNECT, can be abused to send e-mail. It also turns out that in open relays (above), the path that mail took can be detected (the relay server adds information about where it received the message), but in open proxies, it is nearly impossible to detect the true origin of the e-mail; they are thus preferred by spammers. In addition, because the primary purpose of a proxy is for Web access, not e-mail, and because block lists affect only e-mail, there is less incentive for those who run open proxies to close them. Like open relays, those who create open proxies usually do not intend to do so, and substantial amounts of their network bandwidth ends up being used by spammers. (See <http://news.zdnet.co.uk/story/0,,t269-s2122679,00.html> for a good article on open proxies, and see <http://www.kb.cert.org/vuls/id/150227> for a somewhat more technical description of open proxies.)

One very disturbing recent event is the creation of open proxies via viruses or “Trojan horses.” Spammers spread special viruses that, in addition to infecting the user’s machine and spreading themselves, also create open proxies, which the spammer can then abuse to send e-mail. Unlike conventional open proxies that are relatively easy to detect, these special proxies are much harder to detect (they don’t necessarily use the conventional port numbers). This also limits the hope of ever closing all open proxies.

Web Harvesting and Dictionary Attacks

Spammers need to get the e-mail addresses they use from somewhere. One technique is to buy a customer list from another company or individual, possibly one that has not obtained the recipients’ consent. Another technique is Web harvesting: Spammers use software that automatically visits millions of Web pages, searching for anything that looks like an e-mail address, and adds it to the list of addresses to spam

Dictionary attacks are another way that spammers can get e-mail addresses. Spammers simply try sending e-mail to random addresses, such as first names, first names plus digits, common first name-last name combinations, etc. If the mail bounces (an error code is returned), the spammer knows that the address was bad. Any addresses that don’t bounce are presumed to be good. In this way, spammers can guess e-mail addresses that have never been used in any way.

Consumers who receive spam at rarely used accounts sometimes think that an ISP has sold their address, when in fact a spammer has exploited a dictionary attack. Finally, there is even a market in such lists. Some spammers buy lists from others who specialize in any of these techniques.

Industry Initiatives

Filtering techniques and laws are two ways to address the spam problem. There are also important ways that industry can help. One way industry can help is to create programs that establish e-mail best practices. ISPs may then decide to treat mail from senders who participate

in such programs preferentially by allowing such mail to bypass gateway spam filters, whereas mail from nonparticipating senders would be filtered aggressively. (This would not affect the ability of consumers to choose the level of spam protection they desire in their own inboxes.) Second, industry can adopt anti-spoofing measures. Preventing spoofing will not stop any spam, but it will allow safe lists to work more effectively and help prevent fraud.

E-Mail Best-Practice Programs

One exciting possibility for solving the spam problem is that of e-mail best-practice programs. In these programs, legitimate senders would agree to abide by a set of guidelines, procedures, requirements and restrictions established by a self-regulatory program that provides core anti-spam protections for consumers. ISPs would then have an incentive to deliver mail from participating senders without subjecting them to gateway filtering systems. This approach not only reduces the volume of spam but could greatly minimize the problem of false positives (i.e., filtering that mistakenly blocks legitimate mail). Nonparticipating senders would be those who did not wish to agree to these self-regulatory standards. Their mail could be aggressively filtered on the assumption that most legitimate senders would participate in e-mail best-practice programs. If such an e-mail best practice program were widely adopted by both senders and ISPs, it could essentially solve the spam problem. Unfortunately, this will not be simple. Many issues must be resolved. And a legislative “safe harbor” may be required to jump-start such broad adoption.

Propagating Lists: Technical Issues

It is important that some organization keep track of which companies follow the self-regulatory standards and somehow propagate lists of participating senders. There are many technical ways of accomplishing this. A cryptographic certificate can be given to those who comply with the standards and the certificate either stored on the Internet (e.g., in the Domain Name Service [DNS] system) or sent along with each message. Alternatively, recipients can receive lists of those who comply with the standards, or, for each message, can check a central database on the Internet to see if the sender is listed.

None of this is easy. It is important that the lists can be updated quickly. Otherwise spammers can agree to comply with the standards and then send huge amounts of spam before updated lists can be propagated. This means that in some cases, depending on implementation issues, receivers must check updated lists every time they receive a message. Although this is possible, it is quite costly. See <http://www.bondedsender.com/> and <http://eprivacygroup.com/> for examples of ways that such systems might work.

E-Mail Best-Practice Programs: Standards

Will there be one e-mail best-practice program, or many? The prospect of a single worldwide body defining best practices for all e-mail is somewhat worrisome. Who would run this organization? What would people do who disagree with it? How would it address the differing needs of countries around the world?

The other alternative is the creation of various programs for e-mail best-practices. Any company can agree to meet the standards of whichever program it wants, and any enterprise using an

e-mail filter can agree to pass through mail from whichever programs it chooses. This guarantees freedom, but at the cost of complexity: potentially, every consumer must choose which of many programs they trust. If users choose an untrustworthy program, they could unwittingly receive spam. If they choose to trust a program, and over time it ceases to enforce its policies diligently or quickly enough, consumers again risk receiving spam.

One possibility is for the federal government to help define and monitor these programs. If properly defined and implemented, this would solve the problem of whom to trust. Another possibility is that there be potentially hundreds of e-mail best-practice programs but a very small number of rating agencies. These agencies would rate the programs, and the users would choose a rating agency to trust. This structure has several advantages. No single body would control all e-mail; users would be able to exercise choice; users would not need to make difficult decisions. Also, new programs can be created, and as long as the major rating agencies give them good ratings, millions of users would trust mail from those bodies. This would prevent any single organization from gaining control.

Anti-Spoofing

Safe lists, described above, are one of the most common techniques used in anti-spam technologies to help ensure that wanted mail is delivered to recipients. These safe lists usually are used in combination with machine learning filters, matching systems, and challenge-response systems. The problem with safe lists is that there is no way to know whether a sender is who they says they are; the current e-mail protocols are naively based on trust. If spammers can guess commonly safe-listed addresses or can discover users' friends, they spoof these senders' addresses and bypass the filter. Thus for any filtering or challenge-response system to work optimally, this problem must be solved. With industry cooperation, these standards can be changed or extended so that spoofing is difficult or impossible (see for example Hadmut Danisch's proposed RMX record.) Unfortunately, changing standards is hard and time-consuming; this process will take time. (See the Appendix for more information about spoofing.)

Conclusions

Spam is a surprisingly complex problem. No solution is as simple or effective as at first it may seem, and no single solution is likely to solve the problem for all people. Nor are any immediate solutions to the problem likely.

Legislation should be one part of the solution to spam, and one that may have a substantial impact, but it will be difficult to reach the right compromise between laws that are so strict that they do more harm than good, and laws that are so lenient that they are little help.

E-mail best-practice programs should also be pursued, but admittedly they are complex. They raise difficult questions of who will control such programs, what technology will be used and what standards senders will follow. Best-practice programs will probably require creation of whole new types of companies, new technical standards, and updates to at least receiving, and perhaps sending, software. This is a goal that will be difficult and time consuming to achieve, but one worth working toward. If these strategies are successful, they will make a huge dent in the amount of spam that is circulated.

It seems likely that attempts at ephemeral e-mail address systems and micropayment systems will run into too many of the problems described in their respective sections. It also seems likely that filters based on matching systems will be too easy to defeat by randomizing messages. This leaves machine learning filters as the best short-term solution. These systems will filter out most of the spam, with low rates for false positives, and, even with legislation or best-practice programs, will remain an important piece of the solution. If anti-spam legislation is passed, machine learning filters will be used to filter spam from countries without strong anti-spam laws as well as blatantly illegal spam from countries with anti-spam laws. If best-practice organizations are successful, it will still be necessary to aggressively filter mail from nonmembers. Meanwhile, spammers will adapt and attack machine learning systems as we described; it will be the industry's role to counter these attacks, to keep filters working, and to keep e-mail usable.

No single solution will solve the spam problem; no solution will be implemented immediately; and no solution is as easy as it sounds. We believe that with appropriate legislation, industry cooperation and continued improvements in filtering, e-mail will be preserved as a viable communications medium.

Appendices

Technical Details

In this appendix, we provide additional technical details of various anti-spam approaches.

Spoofting

One common trick spammers use is “spoofing.” They may make their mail appear to be from a known source, such as Microsoft or Amazon.com, or from an ISP such as Yahoo! or AOL, where many people have friends. If they can guess addresses on a user's safe list, they may be able to get through spam filters (see the section on safe lists) It is a little surprising that e-mail standards were designed in such a way that it is so easy for people to spoof sending from other systems, and even more surprising that it is so difficult to fix this problem. In this section, we attempt to explain why common solutions do not work and point to some better ideas.

First, it is important to understand that some messages are legitimately spoofed. The one kind of mail that can definitely be detected as spoofed is mail from oneself. For instance, Hotmail can detect when mail allegedly from Hotmail is not really from Hotmail because it originates outside the system, when it should have come directly from Hotmail. There are many examples of such legitimate mail:

- Mailing lists. When a Hotmail user sends mail to a mailing list, such as Yahoo! groups, Yahoo! resends the mail. Yahoo! appropriately sets the “From” line as that of the Hotmail user. Because that user and other Hotmail users are subscribers to the mailing list, Hotmail users receive these spoofed messages.
- Greeting cards. When users use free or paid greeting card systems, the greeting card sender typically sets the “From” line to the user's name and address. Especially in an

age of spam, this is important, because users are unlikely to open mail from people they do not know.

- Online invitations, or “e-vites.”
- Web page recommendations. Some Web pages have a recommendation option to allow users to recommend the page to a friend.
- Some e-mail client software. Some e-mail clients may send mail directly, rather than through a relay. If I have a Hotmail account, I may use it to receive incoming mail but send mail to other Hotmail users directly, rather than through the Hotmail user interface. This mail then has a Hotmail address, although it originates outside of Hotmail. It is spoofed.

Many of these examples are fairly common practice, too common to significantly adversely treat mail that can be detected as spoofed.

Worse, it turns out that in general it is very difficult to detect spoofed mail. The “From” line on e-mail is set to whatever the sender wants to set it to. Similarly, other parts of e-mail, such as the HELO and MAIL FROM lines, are set to whatever the sender sets them to. (These are the name of the sending machine and address that any delivery errors should be sent to.) The only thing that spammers cannot misrepresent is their IP address, which can be reliably detected by the mail recipients.⁵

Now, one might think that Hotmail could check whether mail allegedly from an address, sender@x.com, comes from an x.com IP address. Unfortunately, there is no such list of IP addresses anywhere on the Internet. What one can do is to find the “MX” records for x.com; this is the list of *incoming* IP addresses for x.com, but it is perfectly acceptable, and not uncommon, for large companies to use different computers for their incoming and outgoing mail.

It is also possible to conduct a “reverse IP” lookup to see if the IP address 1.2.3.4 that the receiver got mail from is actually from an x.com machine. Unfortunately, it is possible to lie about reverse IP addresses; whoever controls the IP address 1.2.3.4 can set the reverse IP entry to any value. Also, many reverse IP addresses are not set up correctly. Theoretically, the HELO command in SMTP should be the name of the sending machine, e.g., mail1.x.com, and perhaps

⁵ TCP/IP spoofing is very difficult for spammers. TCP/IP contains a 32-bit sequence number, and recipients do not accept packets unless the sequence number is correct — and there’s only about a one in 4 billion chance of getting that right sequence number by chance. In some cases, if a faker can eavesdrop on a connection, then he can get the right sequence number, but eavesdropping generally requires being on the same subnetwork as the receiver or the alleged sender, and, for spam, this is very unlikely. Cases of incorrectly randomized sequence numbers, which were easy to guess, used to occur more frequently, but this problem has been known for some time and is now rare.

You may also have heard about faking the IP addresses in “received from” headers. When mail is transferred from server to server (between organizations or within), each server tacks on a new received “From” line, with the IP address it received mail from. Of course, each server could modify these and lie about the path of IP addresses taken. However, any server in your own organization won’t lie, and the mail gateway in your organization knows which IP address it received mail from in another organization, the last external hop. This is the one IP address that an external server cannot fake, and the most important one.

one could do a forward address lookup on mail1.x.com and check that the IP addresses match, as well as verifying that the HELO machine is in the same domain as the “From” address (e.g., x.com.) Unfortunately, this fails for a number of reasons. First, it may be acceptable for a machine in the y.com domain to send mail for x.com. For instance, Hotmail machines send mail for MSN users. Second, many people do not correctly configure their HELO settings in their mail servers. Third, as mentioned, spammers can set their HELO settings to anything they want; although one could check for a mismatch between the IP address and HELO, but, because of these common misconfigurations, a mismatch is not conclusive.

There is a solution to most of these problems, but it requires changes in standards as well as wide adoption. The solution is for senders to list all valid outgoing IP addresses for their domain in some way. Most information related to IP addresses is stored in DNS records, making DNS a natural way to go about this. But no DNS record types appropriate for this exist. One possibility is to introduce a new DNS record type (see work by Hadmut Danisch)[1], but processing new types and updating software to be aware of the new types make it a slow process. Another possibility is to use TXT fields of DNS records to store this information, and this method seems more promising. But even here there are issues. Some older DNS implementations are limited in the text record sizes they support, and, for large senders, the list of valid IP addresses may be large.

Even with these changes, the problem of legitimate spoofing remains. Most of this problem can be solved through standards programs. Legitimate spoofers, such as those who run mailing lists, can join the standards program. The standards program will presumably require appropriate headers to indicate the true sender of the mail who has sent mail on behalf of someone else. Spoofed mail from people not in standards programs would be rejected, marked as suspicious or aggressively filtered.

Machine Learning Systems

In this section, we explore the ways that machine learning systems for stopping spam work [4]. The first step for a machine learning system is to obtain training data; that is, messages labeled as good or spam. This training data can come from any of several places: from users who sort all of their e-mail into good mail and spam, by randomly sampling messages, and by asking users to classify them. It can come from a single individual, in which case the filter that is learned is personalized to that individual.

The next step is to extract the “features” in each message. These features include the words in the message and the time the message was sent. The features and the message classifications are then used to train a machine learning system. The system can use any common machine learning technique, such as naive Bayes, neural networks, maximum entropy, decision trees or Bayesian networks. These are all systems that take training data as input and produce a “classifier.” If the features are the words in the message, the system will learn a classifier that associates words such as “click” “here,” “free” and “wet” with spam, as well as associating features such as middle-of-the-night arrival time or the inclusion of images and links. These systems are also trained on good mail. They may learn that the word “wet” is not bad if it is accompanied by the word “weather.” Machine learning systems don’t just learn what distinguishes spam; they also learn what distinguishes good mail.

When a new message comes in, the first step is to find the features in this message, just as was done at training time. These features are then given as input to the classifier, and the classifier produces a verdict, typically either “spam” or “good,” or sometimes a probability or score of “spam” or “good.” Depending on the probability or score, different actions can be taken. If the probability of spam is very high, the system may simply delete the message. If it is moderate, it may go into a special folder.

Bayesian Filters

Bayesian methods have recently become a popular way to filter spam. Typically these methods, such as those described in Paul Graham’s “A Plan for Spam” <http://www.paulgraham.com/spam.html>, are a particular kind of Bayesian method known as naive Bayes. They’re called naive because they make some assumptions that aren’t true in the real world. We have compared our methods to naive Bayes, and ours are definitely better. (In fact, because naive Bayes is one of the easiest methods to implement, we tried these experiments long ago, before they became popular elsewhere.) Bayesian methods are a special kind of probabilistic method, and our methods *are* probabilistic. Strictly speaking, our current methods are not Bayesian.

What exactly does Bayesian mean? Simply put, a Bayesian technique is one in which uncertainty is encoded as probabilities. Any probabilistic spam filter, Bayesian or not, eventually wants to compute the probability $P(\text{spam} \mid \text{message})$, i.e., gauge the likelihood that the message is spam. Bayesian filters typically make use of Bayes’ law:

$$P(\text{spam} \mid \text{message}) = P(\text{message} \mid \text{spam}) \times P(\text{spam}) / P(\text{message})$$

$P(\text{message} \mid \text{spam})$ is the probability of seeing if a particular message, out of the zillions of possible messages, is spam. It’s pretty hard to compute that number, and I’ll get back to it in a minute. $P(\text{spam})$ is just the prior probability that any message is spam; that is, if half your mail is spam overall, then $\frac{1}{2}$. $P(\text{message})$ is the probability of seeing the particular message, whether or not it is spam. This is also very hard to compute, but it turns out not to matter.

We can also compute

$$P(\text{not spam} \mid \text{message}) = P(\text{message} \mid \text{not spam}) \times P(\text{not spam}) / P(\text{message})$$

We can then compare $P(\text{not spam} \mid \text{message})$ and $P(\text{spam} \mid \text{message})$, and make our decision based on which one is larger, and perhaps even how much larger. Notice that we’re going to end up dividing the equation for $P(\text{not spam})$ and the one for $P(\text{spam})$ by $P(\text{message})$, so because they both get divided by the same number, we don’t really care what that number is and we don’t have to actually compute it.

Now here’s the hard part: computing $P(\text{message} \mid \text{spam})$ and $P(\text{message} \mid \text{not spam})$, the probability of seeing the message if it’s spam, and the probability of seeing the message if it’s not spam. This means computing the probability of seeing exactly this message — i.e., if there are 1 billion different spams out there, all equally likely, then $P(\text{message} \mid \text{spam})$ might be one in

a billion. If the message isn't spam, then $P(\text{message} \mid \text{spam})$ is 0. But it's impossible to compute the true probability because we haven't seen every possible spam, and certainly we'll never see every possible message that is not spam. So we have to make approximations. The most common of the many ways to do this is the naive Bayes method, which assumes that every word in the message is independent of every other word. For instance, you assume that the probability of the word "click" occurring in a message is independent of the probability of the word "here" occurring in the same message. It's then easy to go through a whole lot of messages, count up how often "click" or "here" occurs in each message that is spam, and how many spam messages there are overall. Once you've done that, you can get $P(\text{click} \mid \text{spam})$ pretty accurately. The problem with this method is that this assumption is false, and the probabilities you get from something like this don't tend to match the real world very well. If a message has a phrase that occurs more often in spam than in good mail, let's say twice as often, such as "Click here to unsubscribe," you might end up thinking that the message is 16 times ($2 * 2 * 2 * 2$) as likely to be spam, even though really it's perhaps only twice as likely. Of course, we always have to make some assumptions that aren't true in real life because accurately modeling everything in real life is too tough. The question then is just how bad those assumptions are. The naive Bayes assumption isn't terrible, but it's not very good, either.

A different probabilistic approach is, rather than using Bayes law to compute $P(\text{spam} \mid \text{message})$ by using $P(\text{message} \mid \text{spam})$, you just compute $P(\text{spam} \mid \text{message})$ directly. For instance, we can train a neural network to learn this probability. The good thing about that is the fact that the network can learn that words such as "click," "here," "to" and "unsubscribe" tend to occur together, and learn not to overcount them. We aren't talking about exactly what technique we use because we don't want to give spammers information that they can use to get through our filters, but it is similar to a neural network trained to return probabilities.

So Bayesian methods are a special kind of probabilistic technique, and, in general, we really like them. They were a strong contender for Microsoft's spam filtering, though in the end we chose a different technique. The best techniques we've found are probabilistic methods.

You may wonder why Naïve Bayes has become such a popular technique. One reason is that you can get some very good numbers using naive Bayes, over 99%. The problem is, those 99% numbers are usually from people who look at literally thousands of their own e-mail messages, classify them by hand, and use that to train their own personal filter. Not only that, but sometimes they keep looking at every message they receive, spam and good, correcting any errors the filter makes so that it stays good even as mail changes over time; this misses the point of having a spam filter. Worse, those same filters aren't going to work nearly as well for someone else. Different people have very different kinds of e-mail.

We've been focusing our efforts on algorithms that work well out of the box, across lots of different users because we don't think most users want to label thousands of messages before they can get good performance. Of course, MSN also allows users to adapt their filter. If you check and correct all the mistakes your filter makes, the filter in MSN will improve and become even better than one trained with naive Bayes; but it will also work well even if you don't want to take the time to do that.

References

- [1] Dwork, C. and M. Naor, “Pricing via Processing, Or, Combating Junk Mail, Advances in Cryptology,” CRYPTO ’92, Lecture Notes in Computer Science, No. 740, Springer, 1993, pp. 139–147. A more complete version is available at <http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.ps>. You will need a postscript viewer such as Ghostview to view this file.
- [2] Abadi, M.; M. Burrows; M. Manasse; and T. Wobber, “Moderately Hard, Memory-Bound Functions,” Proceedings of the 10th Annual Network and Distributed System Security Symposium, February 2003. Available at <http://research.microsoft.com/aboutmsr/labs/siliconvalley/pubs/memory-final-ndss.pdf>
- [3] Back, Adam, et al. “Hash Cash — A Denial of Service Counter-Measure,” <http://cypherspace.org/hashcash/hashcash/.pdf>. See also <http://cypherspace.org/hashcash>.
- [4] Sahami, M.; S. Dumais; D. Heckerman; and E. Horvitz. A Bayesian Approach to Filtering Junk E-mail. (Postscript file). “AAAI-98 Workshop on Learning for Text Categorization,” July 27, 1998, Madison, Wisconsin.
- [5] Simard, P.Y.; R. Szeliski; J. Benaloh; J. Couvreur; and I. Calinov, “Using Character Recognition and Segmentation to Tell Computer from Humans,” Seventh International Conference on Document Analysis and Recognition, August 2003.
- [6] Gates, Bill. “The Road Ahead,” 1996.